PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-171662
(43)Date of publication of application : 26.06.1998

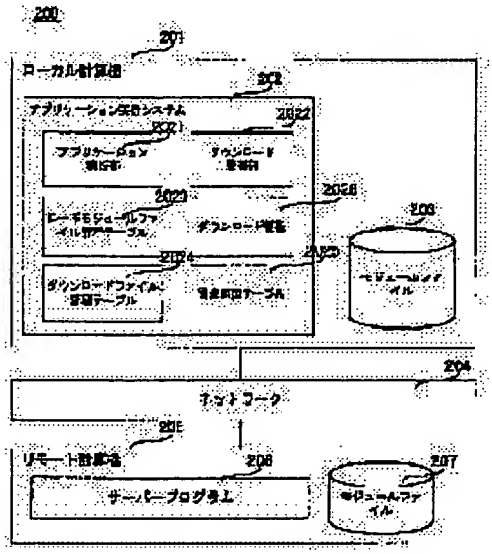
(51)Int.Cl. G06F 9/445
G06F 13/00

(21)Application number : 08-340529
(22)Date of filing : 05.12.1996

(71)Applicant : HITACHI LTD
(72)Inventor : MIZOTE YUJI
TOMINAGA MASASUKE
MASUISHI TETSUYA
HONDA YURI
SHIMAZAKI KOICHI

(54) APPLICATION EXECUTING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce the waiting time of a user in a file download mode by downloading files onto the file system of a local computer from the file system of a remote computer in the order of priority of files and controlling them.
SOLUTION: This application executing system 202 includes an application execution part 2021, a download management part 2022, a load module file management table 2023, a download file management table 2024, the download information 2026 and a priority table 2025. The table 2025 is produced to describe the downloading priority of files forming an application based on the downloaded priority information. Then the files are downloaded onto the file system of a local computer 201 from the file system of a remote computer 205 in the order of priority of files and by making reference to the table 2025 and also to each step. These downloaded files are managed.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平10－171662

(43)公開日　平成10年(1998)６月26日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/445		Ｇ０６Ｆ　9/06	４２０Ｊ
13/00	３５１	13/00	３５１Ｈ

審査請求　未請求　請求項の数３　ＦＤ　（全 22 頁）

(21)出願番号　特願平8－340529

(22)出願日　平成８年(1996)12月５日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　濱手　裕二
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(72)発明者　冨永　雅介
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(72)発明者　増石　哲也
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(74)代理人　弁理士　矢島　保夫

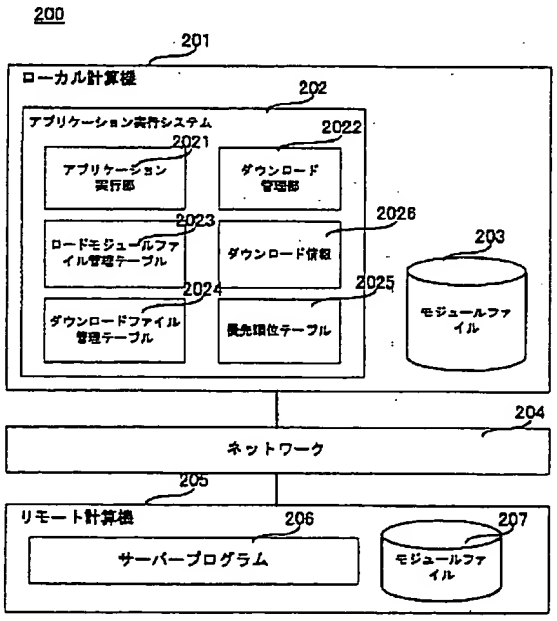
最終頁に続く

(54)【発明の名称】　アプリケーション実行方法

(57)【要約】

【課題】本発明の目的は、アプリケーションを構成するプログラム情報を格納する複数のファイルをリモート計算機からローカル計算機にダウンロードして実行するシステムにおいて、ファイルダウンロードにおけるユーザの待ち時間を削減することである。

【解決手段】特定のファイルのプログラムを実行中に別のファイルが必要になったとき、該ファイルがローカル計算機のファイルシステムに存在しなければ、リモート計算機のファイルシステムからローカル計算機のファイルシステムに該ファイルをダウンロードする処理を行った後、ローカル計算機のファイルシステムにある該ファイルをメモリ中に読み込んで実行する処理と、その処理と並行して行われる処理で、ファイルダウンロードの優先順位を記述する優先順位情報を参照し、その優先順序に基づいて、リモート計算機上のファイルシステムからローカル計算機上のファイルシステムにダウンローデングを行うダウンロード処理を行うようにする。



1

【特許請求の範囲】

【請求項１】複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、

アプリケーションを実行するのに必要なファイルがローカル計算機のファイルシステムに存在するか否かを判定する第１の処理ステップと、

前記第１の処理ステップによりローカル計算機中のファイルシステムに前記ファイルが存在しないことが判明した場合、リモート計算機のファイルシステムからローカル計算機のファイルシステムに前記ファイルをダウンロードする第２の処理ステップと、

前記ファイルをダウンロードしたことを管理する第３の処理ステップと、

前記ファイルをローカル計算機の内部記憶装置に格納するとともに、前記ファイルを内部記憶装置に格納したことを管理する第４の処理ステップと、

前記アプリケーションを構成する複数のファイルのダウンロード処理の優先順位を表す優先順位情報をアプリケーションを構成するファイルの１つ中に含め、前記アプリケーションの実行前に、前記優先順位情報を含むファイルをローカル計算機にダウンロードし、ダウンロードした前記優先順位情報を用いてアプリケーション構成ファイルのダウンロードの優先順位を記述した優先順位テーブルを作成する第５の処理ステップと、

前記第１、第２、第３、および第４の処理ステップと並行して行われる処理であって、前記優先順位テーブルを参照し、その優先順位の順にリモート計算機上のファイルシステムからローカル計算機上のファイルシステムにファイルをダウンロードし、ダウンロードしたファイルを管理する第６の処理ステップとを備えたことを特徴とするアプリケーション実行方法。

【請求項２】前記アプリケーション実行時に、実行対象のファイルを管理する処理ステップと、

前記アプリケーションを構成するファイルの１つ中に、そのアプリケーションを構成する複数ファイル間の依存関係情報を含め、前記依存関係情報および前記実行対象ファイルの管理情報に基づいて、ダウンロードの優先順位をアプリケーションの実行時に動的に決定する処理ステップとをさらに備えたことを特徴とする請求項１に記載のアプリケーション実行方法。

【請求項３】複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、

前記ローカル計算機上で、前記アプリケーションを構成するファイルのうちの１つを内部記憶装置上に格納して実行する実行ステップと、

前記ローカル計算機上で、前記実行ステップとは別プロセスで並行して、前記アプリケーションを構成する他のファイルのダウンロードを順次行うダウンロードステップとを備えたことを特徴とするアプリケーション実行方法。

2

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、アプリケーションを構成するプログラム情報を格納するファイルをリモート計算機のファイルシステムからローカル計算機のファイルシステムにダウンロードして解釈実行するシステム、特に、入出力装置を用いたユーザとの対話的操作を多く含むアプリケーションファイルをダウンロードして実行するシステムにおいて、ファイルダウンローディング時のユーザの待ち時間を削減するアプリケーション実行方法に関する。

【０００２】

【従来の技術】「NetScapeの機能拡張で存在感薄れるＷｉｎ９５」（日経バイト（１９９６年４月、日経ＢＰ社），ｐｐ１６６～１７７）（文献１）に示されているように、近年、世界的規模のネットワーク網であるインターネットとインターネット上の情報開示システムの一つであるWWW（Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ）の発達により、Ｗｅｂサーバ上で開示されたＷｅｂページをクライアント側でＷｅｂブラウザを用いて見ることはもちろんのこと、Ｗｅｂブラウザ上でマウスを用いた簡単な操作でリモート計算機のファイルをクライアントマシンにダウンロードしたり、ダウンロードするファイルに応じたプログラム（ヘルパープログラムやプラグインと呼ぶ）を起動させることにより、リモート計算機のファイルのデータを読み込んでその場でアプリケーションを実行することも可能になった。

【０００３】１つのアプリケーションを構成する複数のファイルをリモート計算機のファイルシステムにおき、それらのファイルをローカル計算機のファイルシステムにダウンロードし、そのファイルをオープンして実行する場合、従来においては、アプリケーション実行前にすべてのファイルをダウンロードするか、実行中にファイルが必要になった時、必要なファイルだけをダウンロードするということが行われていた。

【０００４】また、リモート計算機のファイルシステム上にあるファイルを、ローカル計算機にネットワーク経由で転送し、そのデータを用いてアプリケーションを実

行するアプリケーション実行環境として、Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃが開発したオブジェクト指向型言語Ｊａｖａを用いて作成したアプリケーションをＪａｖａ対応のＷｅｂブラウザ上で実行する場合がある（上記文献１）。ただし、Ｊａｖａは、リモート計算機上のアプリケーションファイルをローカル計算機のファイルシステムに格納してからファイルオープンして実行するのではなく、ローカル計算機のメモリに直接読みこんで実行するものである。Ｊａｖａでは、アプリケーションの実行単位は”アプレット”と呼ばれるプログラム部品である。Ｊａｖａ対応のＷｅｂブラウザでは、Ｗｅｂページ内にサーバー情報を伴ったアプレットを示す記述があると、リモートのサーバー計算機のファイルシステムにあるアプレットファイルをローカル計算機に転送し、ローカルマシン内のＪａｖａ実行システムを用いて、アプレットを実行・表示する。このアプレット自身も、リモート計算機のファイルシステム内の別ファイルに実装された他のＪａｖａプログラム部品を用いて構成することができ、アプレット実行時に、ローカル計算機のＪａｖａ実行システムは、必要な部品のファイルをローカル計算機に新たに転送して実行する。

【０００５】上記従来技術では、リモート計算機からローカル計算機に必要になったファイルだけ転送するので、ファイルのトータルのサイズを一般に小さく保つことができ、一般に時間のかかるネットワークを介してファイルを転送する処理を無駄に行わずに済むという利点があった。

【０００６】

【発明が解決しようとする課題】しかし、上記従来技術では、特に大きなサイズのファイルを転送する場合には、その間ユーザを待たせてしまうという問題があった。また、ＪＡＶＡを用いても、必要なファイルが複数に及ぶ場合はそれらすべてをリモート計算機からローカル計算機に転送するので、結果的に大きなサイズのファイルを転送することになりユーザを待たせることがあった。

【０００７】本発明の目的は、アプリケーションを構成するプログラム情報を格納する複数のファイルをリモート計算機からローカル計算機にダウンロードして実行するシステムにおいて、ファイルダウンロードにおけるユーザの待ち時間を削減することである。

【０００８】

【課題を解決するための手段】上記目的を達成するため、請求項１に係る発明は、複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを

解釈して実行するアプリケーション実行方法において、アプリケーションを実行するのに必要なファイルがローカル計算機のファイルシステムに存在するか否かを判定する第１の処理ステップと、前記第１の処理ステップによりローカル計算機中のファイルシステムに前記ファイルが存在しないことが判明した場合、リモート計算機のファイルシステムからローカル計算機のファイルシステムに前記ファイルをダウンロードする第２の処理ステップと、前記ファイルをダウンロードしたことを管理する第３の処理ステップと、前記ファイルをローカル計算機の内部記憶装置に格納するとともに、前記ファイルを内部記憶装置に格納したことを管理する第４の処理ステップと、前記アプリケーションを構成する複数のファイルのダウンロード処理の優先順位を表す優先順位情報をアプリケーションを構成するファイルの１つ中に含め、前記アプリケーションの実行前に、前記優先順位情報を含むファイルをローカル計算機にダウンロードし、ダウンロードした前記優先順位情報を用いてアプリケーション構成ファイルのダウンロードの優先順位を記述した優先順位テーブルを作成する第５の処理ステップと、前記第１、第２、第３、および第４の処理ステップと並行して行われる処理であって、前記優先順位テーブルを参照し、その優先順位の順にリモート計算機上のファイルシステムからローカル計算機上のファイルシステムにファイルをダウンロードし、ダウンロードしたファイルを管理する第６の処理ステップとを備えたことを特徴とする。

【０００９】請求項２に係る発明は、請求項１において、前記アプリケーション実行時に、実行対象のファイルを管理する処理ステップと、前記アプリケーションを構成するファイルの１つ中に、そのアプリケーションを構成する複数ファイル間の依存関係情報を含め、前記依存関係情報および前記実行対象ファイルの管理情報に基づいて、ダウンロードの優先順位をアプリケーションの実行時に動的に決定する処理ステップとをさらに備えたことを特徴とする。

【００１０】請求項３に係る発明は、複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、前記ローカル計算機上で、前記アプリケーションを構成するファイルのうちの１つを内部記憶装置上に格納して実行する実行ステップと、前記ローカル計算機上で、前記実行ステップとは別プロセスで並行して、前記アプリケーションを構成する他のファイルのダウンロードを順次行うダウンロードステップとを備えたことを

特徴とする。

【００１１】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【００１２】図１は、本発明に係るアプリケーション実行方法を実現するために用いる計算機の構成（１００）の例を示すものである。前記計算機は、内部記憶装置であるメモリ（１０６）、ファイルシステムで管理されるファイルの実体を格納する外部記憶装置（１０３）、アプリケーションの実行処理等を行う中央処理装置（１０４）、ユーザからの入力出力を受けるための入出力装置（１０２）、計算機の処理内容等を表示する表示装置（１０１）、およびネットワークからのデータの入出力を行うネットワーク装置（１０５）から構成され、これらの構成要素は互いにデータを転送するためのバス（１０７）により接続されている。

【００１３】図２は、本発明に係るアプリケーション実行方法を適用したアプリケーション実行システムの構成およびその動作環境（２００）を示す。本アプリケーション実行システム（２０２）は、アプリケーション実行部（２０２１）、ダウンロード管理部（２０２２）、ロードモジュールファイル管理テーブル（２０２３）、ダウンロードファイル管理テーブル（２０２４）、ダウンロード情報（２０２６）、および優先順位テーブル（２０２５）を備えている。

【００１４】次に、本アプリケーション実行システムの動作環境について述べる。動作環境としては、「従来の技術」で述べた従来技術であるWWWを利用する。計算機（２０１）は、アプリケーション実行システム（２０２）が実行される計算機であり、これをローカル計算機と呼ぶ。計算機（２０５）は、ネットワーク（２０４）を介してローカル計算機（２０１）と接続されている。前記計算機（２０５）をリモート計算機と呼ぶ。ローカル計算機（２０１）とリモート計算機（２０５）は図１に示した構成を有する。リモート計算機（２０５）上では、サーバープログラム（２０６）と呼ばれるプログラムが実行されており、公知のネットワーク技術を利用して、ローカル計算機（２０１）上で実行されているアプリケーション実行システム（２０２）が、リモート計算機（２０５）上のサーバープログラム（２０６）に対してリモート計算機（２０５）上のファイルシステム（２０７）中にあるファイルの転送要求を行い、その要求に対して、リモート計算機（２０５）は要求対象となっているファイルをローカル計算機（２０１）に転送し、ローカル計算機（２０１）のファイルシステム（２０３）に格納する。

【００１５】本実施形態では、一つのアプリケーションは、２つ以上の複数のファイルから構成される。これらのファイルをモジュールファイルと呼ぶ。モジュールファイルには、アプリケーションを構成するプログラムと

該プログラム実行に必要なデータが格納されている。特に、アプリケーションの実行開始時に実行されるプログラムを含むファイルを先頭モジュールファイルと呼ぶ。モジュールファイルは、ローカル計算機（２０１）のファイルシステム（２０３）、または、リモート計算機（２０５）のファイルシステム（２０７）に存在する。

【００１６】本実施形態では、ローカル計算機（２０１）がアプリケーションの実行時に必要とするファイルをリモート計算機（２０５）に対して転送要求を行い、ローカル計算機（２０１）が前記ファイルを取得し、アプリケーションの実行を行う。

【００１７】図３は、ロードモジュールファイル管理テーブル（２０２３）のデータ構造（３００）を示すものである。ロードモジュールファイル管理テーブル（２０２３）は、メモリ上にロードされたモジュールファイルを管理するためのテーブルであり、ロードモジュールファイルエントリ（３０１）と呼ばれるデータ構造を要素として持つ配列である。ロードモジュールファイルエントリ（３０１）は、モジュールファイル名称と呼ばれる文字列（３０１１）、およびロードされたモジュールファイルのメモリ中の先頭アドレス（３０１２）から構成されるデータ構造である。

【００１８】図４は、ダウンロードファイル管理テーブル（２０２４）のデータ構造（４００）を示すものである。ダウンロードファイル管理テーブル（２０２４）は、ダウンロードされたモジュールファイルを管理するためのテーブルであり、ダウンロードファイルエントリ（４０１）と呼ばれるデータ構造を要素として持つ配列である。ダウンロードファイルエントリ（４０１）は、モジュールファイル名称と呼ばれる文字列（４０１１）、およびダウンロード状態（４０１２）から構成されるデータ構造である。ダウンロード状態（４０１２）は、整数型のデータ構造であり、対応するモジュールファイルの状態として、ＤＯＷＮＬＯＡＤＥＤまたはＮＯＴ＿ＤＯＷＮＬＯＡＤの２つの値の何れかをとる。

【００１９】図５は、ダウンロード情報（２０２６）のデータ構造（５００）を示すものである。ダウンロード情報（２０２６）は、バックグラウンドでダウンロードを行う際（詳しくは後述するが、本アプリケーション実行システムでは、アプリケーションの実行処理を行うメイン処理プロセスのバックグラウンドでダウンロードを行うダウンロードプロセスを実行するようになっている）のスレッドの制御のために用いるテーブルである。ダウンロード情報（２０２６）は、スレッドフラグ（５０１）、カレントダウンロードファイルインデックス（５０２）、リモート計算機のネットワークアドレス（５０３）、モジュールファイルのリモート計算機上での格納先ディレクトリ（５０４）、およびインストールディレクトリ（５０５）からなる。

【００２０】スレッドフラグ（５０１）は、整数型のデ

ータ型で、値としてＡＰＰＬＩＣＡＴＩＯＮ＿ＱＵＩＴ、ＤＯＷＮＬＯＡＤ＿ＱＵＩＴ、ＡＰＰＬＩＣＡＴＩＯＮ＿ＲＵＮの３つの値をとる。スレッドフラグ（５０１）は、アプリケーションを実行するプロセスとダウンロードを実行するプロセスとの間で同期をとるために用いるフラグである。カレントダウンロードファイルインデックス（５０２）は、整数型のデータ構造であり、優先順位テーブル（６００）中のあるエントリのインデックス値（どこまでダウンロードしたかを示すインデックス値）を保持する。リモート計算機のネットワークアドレス（５０３）は、アプリケーション実行システム（２０２）が実行の対象とするアプリケーションを構成するファイルをファイルシステム中に保持するリモート計算機（２０７）のネットワークの上のアドレスを保持するデータ構造である。モジュールファイルのリモート計算機上での格納先ディレクトリ（５０４）は、前記リモート計算機（２０７）のファイルシステム（２０７）に格納されているアプリケーションのモジュールファイルの格納先ディレクトリパスを保持するデータ構造である。インストールディレクトリ（５０５）は、モジュールファイルのローカル計算機（２０１）上での格納先ディレクトリである。

【００２１】図６は、優先順位テーブル（２０２５）のデータ構造（６００）を示すものである。優先順位テーブル（２０２５）は、１つのアプリケーション実行ごとに用意されるテーブルであり、そのアプリケーションを構成する複数のモジュールファイルをダウンロードする際の優先順位を格納したテーブルである。優先順位テーブル（２０２５）は、優先順位テーブルエントリ（６０１）と呼ぶデータ構造を要素として持つ配列である。優先順位テーブルエントリ（６０１）は、優先順位を表すインデックスと呼ぶ整数型のデータ構造（６０１１）およびモジュールファイル名称と呼ぶ文字列（６０１２）から構成されるデータ構造である。

【００２２】図７は、本実施の形態で述べるアプリケーション実行システムが実行対象とするプログラムの構造（７００）である。先に述べたようにアプリケーション（７００）は、１つ以上の複数のモジュールファイルから構成される。先頭モジュールファイル（７０１）は、イベントマップ（７０２）、処理ブロックの配列（７０３）、プログラム実行に必要なデータ（７０４）、リモート計算機（２０５）のネットワーク上の位置情報（７０５）、モジュールファイルのリモート計算機上での格納先ディレクトリパス情報（７０６）、モジュールファイル優先順位情報（７０７）、およびアプリケーションのモジュールファイル構成（７０８）から構成される。モジュールファイル（７０９）は、処理ブロックの配列（７１０）およびアプリケーションの実行に必要なデータ（７１１）から構成される。

【００２３】処理ブロックは、プログラムを構成する手

続きであり、アプリケーション実行部（２０２１）により解釈実行可能な形態で、ファイルに格納されたもののことである。リモート計算機のネットワーク上の位置情報（７０５）は、リモート計算機（２０５）のネットワーク上の位置情報を保持する領域である。モジュールファイルのリモート計算機（２０５）上での格納先ディレクトリパス情報（７０６）は、アプリケーション実行システム（２０２）の実行対象となるアプリケーションを構成するモジュールファイルのリモート計算機（２０５）のファイルシステム（２０７）上の格納先ディレクトリパス情報を保持する領域である。モジュールファイル構成（７０８）は、このアプリケーションを構成するモジュールファイル名称の配列である。

【００２４】図８は、モジュールファイル優先順位情報（７０７）のデータ構造（８００）を示す図である。モジュールファイル優先順位情報（８００）は、モジュールファイルをリモート計算機（２０５）からダウンロードする際の優先順位情報を保持する領域であり、優先順位エントリ（８０１）の配列である。優先順位エントリ（８０１）は、整数型の優先順位（８０１１）およびモジュールファイル名称（８０１２）である文字列から構成される。優先順位（８０１１）は、１から順にモジュールファイル数だけ割り振られ、同じ優先順位を持つモジュールファイルはない。

【００２５】次にイベントマップについて説明する。本実施形態のアプリケーション実行部（２０２１）のアプリケーションの実行方式は、イベント駆動型と呼ばれるものであるとする。イベント駆動型のアプリケーション実行部（２０２１）は、以下のデータを管理する。すなわち、イベントキュー、およびイベントマップの先頭アドレスである。

【００２６】イベントとは、数値型のデータである。イベントキューは、前記イベントのリストであり、２つの関数、すなわちＰｏｓｔＥｖｅｎｔ関数およびＧｅｔＥｖｅｎｔ関数により、操作されるものとする。ＰｏｓｔＥｖｅｎｔ関数は、イベントを引数としてもつ関数であり、イベントキューの最後に引数イベントを加える処理を行い、戻り値はない。入出力装置（１０２）で発生する入出力イベントはオペレーティング・システムからＰｏｓｔＥｖｅｎｔ関数を通して、アプリケーションに通知される。ＧｅｔＥｖｅｎｔ関数は、引数なしの関数であり、戻り値として、イベントキューの先頭のイベントを返し、イベントキューの先頭イベントを削除する。

【００２７】イベントキューが長さ０のとき、ＧｅｔＥｖｅｎｔ関数は、制御を呼び出し側に戻さず、呼び出し側のスレッドの実行をアイドル状態にする。別スレッドあるいはオペレーティング・システムが、前記イベントキューに対して、ＰｏｓｔＥｖｅｎｔを実行し、イベントキューが長さ１以上になった時点で、前記ＧｅｔＥｖｅｎｔ関数の呼び出しのアイドル状態を開放し、実行を

再開するものとする。

【0028】イベントマップの先頭アドレスは、イベントマップ（図9で詳述する）と呼ぶデータのメモリ中の先頭アドレスを保持するデータ構造である。

【0029】図9は、イベントマップ（702）のデータ構造（900）を示したものである。イベントマップ（900）は、イベント管理情報（901）と呼ばれるデータ構造を要素として持つ配列である。イベント管理情報（901）は、イベント（9011）、当該イベントに対応する処理ブロックのモジュール内での相対アドレス（9012）、当該処理ブロックを含むモジュールファイル名称（9013）からなるデータ構造である。

【0030】イベントマップ（900）は、DispachMap関数を通して操作（参照）される。DispachMap関数は、引数としてイベントをとり、「イベントマップの先頭アドレス」で指定されるイベントマップ（900）を走査して、引数で指定されたイベントに対応する処理ブロックのアドレス（9012）と当該処理ブロックを含むモジュール名称（9013）の対をDispachMap関数の戻り値として返す。

【0031】アプリケーションの特別なイベントとして、QUITイベントとSTARTイベントがある。STARTイベントに対してアプリケーションの実行開始処理ブロックが、QUITイベントに対してアプリケーションの終了処理ブロックが、それぞれイベントマップ（900）中で対応付けられる。イベント駆動型のアプリケーションの実行の終了は、プログラム実行中にQUITイベントが発生することにより行われる。

【0032】以上が、本実施形態で述べるプログラムの構成の一例である。これはあくまで、以降で述べる本発明に係るプログラム実行方法の実施形態を具体的に記述するために便宜的に定義したものであることをここで断っておく。

【0033】次に、先頭モジュールファイル（701）のダウンロードとアプリケーション実行システム（202）の起動について、図10および図20を用いて述べる。本実施形態においては、先頭モジュールファイル（701）のダウンロードと、これまで説明したアプリケーション実行システム（202）の起動は、「従来の技術」で述べた通り、従来技術であるWWWとWebブラウザ（1002）の機能を用いて行われる。

【0034】図20は、HTMLのスクリプト例（2000）である。図10において、Webブラウザを構成する処理（10021）～（10025）および（1002）は、Webブラウザを説明するために便宜的にここで定義したものであり、実際のWebブラウザのシステム構成と必ずしも一致するものではないことを断っておく。Webブラウザ（1002）は、HTML表示処理（10022）により、HTML（Hyper Text Markup Language）スクリプトを読み込んでブラウザ画面（1001）に文字、図形、画像等を表示する。HTMLの詳細については、例えば、文献：“OpenDesign、No. 13、1996年4月、CQ出版社、p4からp13、p40からp59”で説明されている。

【0035】図20のHTMLスクリプト（2000）において、文字、図形、画像等に対してハイパーリンク先であるURL情報が対応づけられているとき、これをアンカーと呼ぶ。HTMLスクリプト（2000）の例では、（2001）に示すように、図10のブラウザ画面（1001）上の文字“アプリケーション1”はアンカーであり、URL"http://zzz.hitachi.co.jp/App1/apl1.fst"が対応づけられている。apl1.fstが、アプリケーション実行システム（202）が実行対象とするアプリケーションの先頭モジュールファイルのファイル名称である。

【0036】入力処理（10021）により、ブラウザ画面（1001）上で文字列“アプリケーション1”が入出力装置（102）を通してクリックされたことが判明すると、ファイル転送要求処理（10023）は、URL情報を用いて、本例ではネットワーク上の位置情報が“zzz.hitachi.co.jp”であるリモート計算機（205）に対して、通信プロトコル“HTTP”で、ファイル"/App1/apl1.fst"のファイル転送を要求する。前記転送要求に対して、リモート計算機（205）上のサーバープログラム（206）は、要求された前記ファイル“apl1.fst”をローカル計算機（201）のファイルシステム（203）に転送する。本実施形態においては、先頭モジュールファイル（701）はこのようにして、ローカル計算機（201）のファイルシステム（203）に転送される。さらに、Webブラウザ（1002）の結果解析処理（10024）は、ダウンロードされたファイルのヘッダ情報を解析し、転送されたファイルの種別を特定し、その後の処理を振り分ける。例えば、ファイルの種別がHTMLファイルである場合には、HTML表示処理（10022）によりWebブラウザ画面（1001）に表示を行わせる。あらかじめ、ブラウザプロセス（1002）は、ファイル種別とそのファイルを処理することができるアプリケーションを対応付けた「AP起動対応表」（10025）を持っている。結果解析処理（10024）は、ファイルダウンロード後、このAP起動表（10025）を参照して、転送されたファイルのファイル種別に対応するアプリケーションを起動する。本実施形態では、先頭モジュールファイル（701）のダウンロード後、そのダウンロードした前記先頭モジュールを入力として、アプリケーション実行部（2021）のアプリケーションの実行処理を開始する。

【0037】ローカル計算機（201）のアプリケーシ

ョン実行部（２０２１）のアプリケーションの実行処理（１１００）とダウンロード管理部（２０２２）の実行（１７００）とは、別スレッドとして、並行して実行される。本発明の実施形態は、マルチスレッドが可能で、かつスレッドの優先順位の指定ができるオペレーティング・システム上で実現されるものとする。

【００３８】また、アプリケーションの実行処理（１１００）のスレッド（メイン処理スレッドと呼ぶ）とダウンロード管理部の実行（１７００）のスレッド（ダウンロードスレッドと呼ぶ）とは、アプリケーションの実行の終了のタイミングにおいて、２つのスレッドの終了処理に関して同期をとる必要がある。そこで、前記２つのスレッド間で、前記スレッドフラグ（５０１）と呼ぶデータを通して、同期をとる。

【００３９】図１１は、アプリケーション実行部（２０２１）によるアプリケーションの実行処理（１１００）のフローチャートを示している。本処理は、アプリケーション実行部（２０２１）の起動時の入力である先頭モジュールファイルを入力として持ち、本処理内のローカル変数として、変数ＥＶＥＮＴ、変数ＰＲＯＣ、および変数ＭＯＤＵＬを持つ。

【００４０】まずステップ（１１０１）で、アプリケーション実行部（２０２１）の実行開始処理（図１２の１２００）を行う。次にステップ（１１０２）で、ＳＴＡＲＴイベントを引数としてＰｏｓｔＥｖｅｎｔを実行し、当該アプリケーションのイベントキューの先頭にＳＴＡＲＴイベントを挿入する。ステップ（１１０３）で、ＧｅｔＥｖｅｎｔ関数の実行により、イベントキューの先頭イベントを取得し、変数ＥＶＥＮＴに格納する。判断（１１０４）で、変数ＥＶＥＮＴの値がＱＵＩＴイベントであるか判定する。この判定結果が真である場合、ステップ（１１０７）に飛ぶ。この判定結果が偽である場合は、ステップ（１１０５）に飛ぶ。ステップ（１１０５）では、変数ＥＶＥＮＴの値を引数として、ＤｉｓｐａｃｈＭａｐ関数を実行し、変数ＥＶＥＮＴの値に対応する処理ブロックのモジュール内の相対アドレスと、当該処理ブロック含むモジュールファイル名を取得する。前記処理ブロックのアドレスを変数ＰＲＯＣに、モジュールファイル名を変数ＭＯＤＵＬに格納する。次にステップ（１１０６）で、変数ＰＲＯＣの値と変数ＭＯＤＵＬの値を入力として、アプリケーション実行部（２０２１）のプログラム処理ブロックの実行処理（図１４の１４００）を実行する。前記処理ブロックの実行の終了後、ステップ（１１０３）に戻る。

【００４１】ステップ（１１０７）では、アプリケーションの終了処理（１５００）を行う。ステップ（１１０７）の後、本処理を終了する。

【００４２】図１２は、ステップ（１１０１）で実行するアプリケーション実行部（２０２１）の実行開始処理（１２００）のフローチャートを示している。この処理は、先頭モジュールファイルを入力として持つ。本処理内のローカル変数としてｎを持つ。

【００４３】ステップ（１２０１）で、本アプリケーション実行のイベントキューを生成する。この時点でイベントキューの長さは０である。ステップ（１２０２）で、入力である先頭モジュールファイル（７０１）をロードする。ロードされた先頭モジュールファイル（７０１）のイベントマップ（７０２）のメモリ上のアドレスをイベントマップの先頭アドレスに設定する。これにより、先頭モジュールファイル（７０１）中に含まれるイベントマップ（７０２）が、関数ＤｉｓｐａｃｈＭａｐを通してアクセス可能となる。

【００４４】次に、ステップ（１２０３）で、ロードされた先頭モジュールファイル（７０１）のサーバー位置情報（７０５）、アプリケーションのリモート計算機上での格納先ディレクトリパス（７０６）を、それぞれ、ダウンロード情報（２０２６、図５の５００）のリモート計算機のネットワーク位置情報（５０３）、アプリケーションのリモート計算機上での格納先ディレクトリパス（５０４）に格納する。また、ダウンロード情報（５００）のカレントダウンロードファイルインデックス（５０２）に１を設定する。さらにダウンロード情報（５００）のスレッドフラグ（５０１）をＡＰＰＬＩＣＡＴＩＯＮ＿ＲＵＮに設定する。ダイアログボックスを表示し、ユーザにインストールディレクトリパス（５０５）の値を設定してもらう。次に、アプリケーションモジュールファイル構成（７０８）の長さで、ダウンロードファイル管理テーブル（２０２４、図４の４００）、ロードモジュールファイル管理テーブル（２０２３、図３の３００）を初期化する。具体的には、まず、各テーブルの確保した配列領域（当該アプリケーションを構成するファイルモジュールの数分のデータエントリが確保される）を０で初期化する。ダウンロードファイル管理テーブル（４００）の１番目の要素のモジュールファイル名称（４０１１）に、先頭モジュールファイル（７０１）のファイル名称を設定する。ロードモジュールファイル管理テーブル（３００）の１番目の要素のモジュールファイル名称（３０１１）に、先頭モジュールファイル（７０１）のファイル名称を設定する。ダウンロードファイル管理テーブル（４００）の１番目の要素のダウンロード状態（４０１２）には、ＤＯＷＮＬＯＡＤＥＤを設定する。ロードモジュールファイル管理テーブル（３００）の１番目の要素の先頭アドレス（３０１２）には、ロードされた先頭モジュールファイル（７０１）の先頭アドレスを設定する。ローカル変数ｎに２を設定する。以上のステップ（１２０３）の処理の後、ステップ（１２０４）に進む。

【００４５】判断（１２０４）で、式「ｎ＜＝モジュールファイル構成（７０８）の長さ」を評価し、その結果が真である場合は、ステップ（１２０５）に飛ぶ。前記

評価結果が偽の場合は、ステップ（１２０８）に飛ぶ。ステップ（１２０５）では、ロードモジュールファイル管理テーブル（３００）のｎ番目の要素のモジュールファイル名称（３０１１）に、先頭モジュールファイル（７０１）のモジュールファイル構成（７０８）のｎ番目の要素のモジュールファイル名称を設定する。ステップ（１２０６）では、ダウンロードファイル管理テーブル（４００）のｎ番目の要素のモジュールファイル名称（４０１１）に、先頭モジュールファイル（７０１）のモジュールファイル構成（７０８）のｎ番目の要素のモジュールファイル名称を設定する。ステップ（１２０７）で、式「ｎ：＝ｎ＋１」を実行する。ステップ（１２０８）では、優先順位テーブルの生成処理（図１３の１３００）を行い、後述する処理（図１７の１７００）を行うダウンロードスレッドを生成する。

【００４６】図１３は、ステップ（１２０８）で実行する優先順位テーブルの生成処理（１３００）のフローチャートを示す図である。本処理は、入力として先頭モジュールファイルを持ち、本処理内のローカル変数としてｎを持つ。ステップ（１３０１）で、優先順位テーブル（２０２５、図６の６００）を、入力である先頭モジュールファイル（７０１）のモジュールファイル構成（７０８）の長さで初期化する。次にｎに１を設定する。判断（１３０２）で、式「ｎ＜＝モジュールファイル構成（７０８）の長さ」を評価し、その評価結果が真である場合、ステップ（１３０３）に飛ぶ。前記評価結果が偽である場合、本処理を終了する。ステップ（１３０３）で、優先順位テーブル（６００）のｎ番目の要素のインデックス（６０１１）に、優先順位情報（７０７）のｎ番目の要素の優先順位（８０１１）を設定する。ステップ（１３０４）で、優先順位テーブル（６００）のｎ番目の要素のモジュールファイル名称（６０１２）に、優先順位情報（７０７）のｎ番目の要素のモジュールファイル名称（８０１２）を設定する。ステップ（１３０５）で、式「ｎ＝ｎ＋１」を実行する。ステップ（１３０５）の後、ステップ（１３０２）へ戻る。

【００４７】図１４は、ステップ（１１０６）のアプリケーション実行部（２０２１）によるプログラム処理ブロックの実行処理（１４００）のフローチャートを示すものである。本処理は、実行対象となる処理ブロックのモジュール内の相対アドレスと前記処理ブロックを含むモジュール名を入力として持ち、本処理内のローカル変数として、変数ＭＯＤＵＬＴＯＰを持つ。

【００４８】ステップ（１４０１）で、入力であるモジュール名を入力としてダウンロード処理（図１９の１９００）を実行する。なお、ここでいうダウンロード処理はダウンロード命令の発行を行うものである（実際のダウンロードはダウンロードスレッドで実行する）。ステップ（１４０２）で、入力であるモジュールファイル名称を入力として、アプリケーション実行部（２０２１）によるロード処理（図１５の１５００）を行う。前記処理のリターン値を変数ＭＯＤＵＬＴＯＰに格納する。ステップ（１４０３）で、変数ＭＯＤＵＬＴＯＰには、入力であるモジュールファイルのメモリ中の先頭アドレスが格納されている。前記アドレスをベースアドレス、入力である処理ブロックのアドレスをオフセットアドレスとして、メモリ中の処理ブロックの実アドレスを計算し、アプリケーション実行部（２０２１）が処理ブロックを解釈・実行する。以上で本処理を終了する。

【００４９】図１５は、ステップ（１４０２）のアプリケーション実行部（２０２１）によるモジュールファイルのロード処理（１５００）のフローチャートを示すものである。本処理は、入力としてモジュールファイル名称を持ち、出力としてメモリのアドレスを返す。また、本処理は、本処理内のローカル変数として、変数ＭＯＤＵＬＴＯＰを持つ。

【００５０】判断（１５０１）で、入力であるモジュールファイルがすでにダウンロードされているかを調べる。これは、ダウンロードファイル管理テーブル（２０２４、図４の４００）を走査し、入力であるモジュールファイルに対応するダウンロードファイルエントリ（４０１）のダウンロード状態（４０１２）がＤＯＷＮＬＯＡＤＥＤと等しくなっているかを判定することにより行う。もし、前記判定で真となっている場合、入力であるモジュールファイルはダウンロードされている。前記判定で偽となっている場合、入力であるモジュールファイルはダウンロードされていない。ダウンロードされている場合は、ステップ（１５０２）に飛ぶ。ダウンロードされていない場合は、ステップ（１５０３）に飛ぶ。ステップ（１５０３）では、入力であるモジュールファイル名称を入力としてダウンロード処理（図１９の１９００）を実行する。ステップ（１５０３）の後、ステップ（１５０４）に進む。

【００５１】判断（１５０２）では、入力であるモジュールファイルがメモリ中にロードされているか判定する。この判定は、ロードモジュールファイル管理テーブル（２０２３、図３の３００）を走査し、入力であるモジュールファイルに対応するロードモジュールファイルエントリ（３０１）の先頭アドレス（３０１２）が０以外になっている場合、前記判定結果を真とし、それ以外を偽とする。この判定結果が真の場合、変数ＭＯＤＵＬＴＯＰにロードモジュールファイルエントリ（３０１）の先頭アドレス（３０１２）を設定して、ステップ（１５０５）に飛ぶ。ステップ（１５０２）の判定結果が偽である場合は、ステップ（１５０４）に飛ぶ。

【００５２】ステップ（１５０４）では、入力であるモジュールファイルをメモリ中にロードし、その先頭アドレスを変数ＭＯＤＵＬＴＯＰに格納する。ロードモジュールファイル管理テーブル（２０２３、図３の３００）を走査し、入力であるモジュールファイル名称に対応す

るロードモジュールファイルエントリ（301）を取得する。前記ロードモジュールファイルエントリ（301）の先頭アドレス（3012）に変数MODULTOPの値を格納する。ステップ（1505）では、本処理の出力として、変数MODULTOPの値を返す。以上で本処理を終了する。

【0053】図16は、ステップ（1107）のアプリケーション実行部（2021）によるアプリケーション終了処理（1600）のフローチャートである。判断（1601）で、スレッドフラグ（501）がDOWNLOAD_QUITになっているかを判定する。この結果が真である場合は、ステップ（1604）に飛ぶ。前記判定の結果が偽である場合は、ステップ（1602）に飛ぶ。ステップ（1602）で、スレッドフラグ（501）をAPPLICATION_QUITに設定する。ステップ（1603）で、ダウンロードスレッドが終了するまで、メイン処理のスレッドを「アイドル状態」にし、ダウンロードスレッドの終了後、以降の処理が再開されるようにスレッド制御を行うようオペレーティング：システムに要求する。ステップ（1604）で、アプリケーションの実行を終了する。以上で本処理を終了する。

【0054】図17は、第二のスレッドであるダウンロードスレッド（図12のステップ（1208）で生成）で実行される処理（1700）のフローチャートである。ダウンロードスレッドで行われる処理は、メイン処理スレッドと並行して行われる。ただし、ダウンロードスレッドは、メイン処理スレッドよりスレッドの優先順位を低く設定して実行される。本処理は、入力はないが、本処理のローカル変数としてＦＩＬＥを持つ。

【0055】判断（1701）で、スレッドフラグ（501）にAPPLICATION_QUITが設定されているか否か判定する。前記判定の結果が真である場合、ステップ（1705）に飛ぶ。前記判断の結果が偽である場合、判断（1702）に飛ぶ。判断（1702）で、すべてのファイルがダウンロードされているか否かの判定を行う。この判定では、ダウンロードファイル管理テーブル（400）中の各エントリ（401）を走査し、すべてのエントリ（401）のダウンロード状態（4012）がDOWNLOADEDになっている場合、前記判定結果を真とし、それ以外の場合、偽とする。本判断の結果が真の場合は、ステップ（1705）に飛ぶ。本判断の結果が偽の場合、ステップ（1703）に飛ぶ。ステップ（1703）で、ダウンロードファイル決定処理（図18の1800）を実行する。その結果を変数ＦＩＬＥに格納する。ステップ（1704）で、変数ＦＩＬＥの値を入力として、ダウンロード処理（図19の1900）を実行する。ステップ（1704）の後、ステップ（1701）に戻る。ステップ（1705）では、スレッドフラグ（501）をDOWNLOAD_QUITに設定し、ダウンロードスレッドを終了する。

【0056】図18は、ダウンロードスレッドのステップ（1703）で実行されるダウンロードファイル決定処理（1800）のフローチャートである。本処理は、入力はないが、出力としてダウンロードすべきモジュールファイル名を返す。また、本処理は、本処理内のローカル変数としてＦＩＬＥを持つ。

【0057】ステップ（1801）で、変数ＦＩＬＥに空文字列を設定する。ステップ（1802）で、カレントダウンロードファイルインデックス（502）の値を1だけインクリメントする。判断（1803）で、カレントダウンロードファイルインデックス（502）に対応するインデックス値を持つ優先順位テーブル（600）中のテーブルエントリ（601）が存在するか否かの判定を行う。前記判定の結果が真（存在する）である場合、ステップ（1804）に飛ぶ。前記判定が偽である場合は、ステップ（1811）に飛ぶ。

【0058】ステップ（1804）で、カレントダウンロードファイルインデックス（502）に対応する優先順位テーブル（600）中のテーブルエントリ（601）を取得する。ステップ（1805）で、前記テーブルエントリ（601）のモジュールファイルのダウンロード状態（4012）を取得する。判断（1806）で、前記ダウンロード状態がDOWNLOADEDであるか否か判定する。この判定結果が真である場合、ステップ（1802）に飛ぶ。前記判定結果が偽である場合、ステップ（1807）に飛ぶ。ステップ（1807）で、前記エントリのモジュールファイル名称の項目を参照し、その値を変数ＦＩＬＥに設定する。ステップ（1808）で、変数ＦＩＬＥの値を本処理の出力として返し、本処理を終了する。

【0059】図19は、ダウンロード処理（1900）のフローチャートである。本処理は、入力としてモジュールファイル名称をとる。ダウンロード処理（1900）は、図14のステップ（1401）、図15のステップ（1503）、およびダウンロードスレッドのステップ（1704）で実行される処理である。

【0060】判断（1901）で、入力であるモジュール名のファイルがローカルファイルシステム（203）中のインストールディレクトリパス（505）下にあるか検索し、存在する場合は、ステップ（1904）に飛ぶ。もし存在しなければ、ステップ（1902）に飛ぶ。ステップ（1902）で、リモート計算機のネットワーク上の位置情報（503）で特定されるリモート計算機（205）に対して、モジュールファイルのリモート計算機上での格納先ディレクトリ（504）で指定されるリモート計算機のファイルシステム上のディレクトリ下にあるファイルで、入力であるモジュールファイル名称と同じファイル名称を持つファイルの転送要求を行

う。ステップ（１９０３）で、転送されたファイルを、入力であるモジュールファイル名称と同じファイル名称で、インストールディレクトリパス（５０５）で示されるローカル計算機のファイルシステムのディレクトリの下に格納する。ステップ（１９０４）で、ダウンロードファイル管理テーブル（２０２４、図４の４００）から、入力であるモジュールファイル名称に対応するテーブルエントリ（４０１）を取得し、そのテーブルエントリ（４０１）のダウンロード状態（４０１２）をＤＯＷＮＬＯＡＥＤに設定する。

【００６１】以上で、本発明の一実施形態であるアプリケーション実行システム（２０２）を説明した。上記第１の実施形態によれば、別スレッドでもモジュールファイルのダウンロードを行うので、ユーザとの対話的操作を多く含むアプリケーションファイルをダウンローディングして実行する場合、ユーザ入力の待ち時間を利用してダウンロードを行うことができ、必要になったときには既にダウンロードが終了しているというようにできる。したがって、ファイルダウンローディング時のユーザの待ち時間を削減してユーザの快適さを向上させることができる。

【００６２】次に、先に述べたアプリケーション実行システム（２０２）を拡張した第２の実施の形態について述べる。上記第１の実施の形態では、ダウンロードする順序はスタティックに決定されていた。第２の実施の形態では、メモリに最新にロードされたモジュールファイルに依存するモジュールファイルを、次に実行される可能性の高いモジュールとして、早めにダウンロードするようにした。第２の実施の形態の構成や処理は、第１の実施の形態と同様であるので、以下では異なる部分のみ説明する。特に、先頭モジュールファイル（７０１）、ダウンロード情報のデータ構造（５００）、処理（１５００）、処理（１８００）の変更点および新規に追加したデータ構造（２１００）の説明を行う。

【００６３】まず、先頭モジュールファイル（７０１）に、モジュールファイル間依存関係情報を付加する。モジュールファイル間依存関係情報については、図２１で詳述する。

【００６４】図２１に、モジュールファイル間依存関係情報のデータ構造（２１００）を示す。モジュールファイル間依存関係情報は、依存情報エントリ（２１０１）と呼ぶデータ構造の配列である。依存情報エントリ（２１０１）は、モジュールファイル名称（２１０１１）および依存する他のモジュールファイル名称配列（２１０１２）から構成される。依存する他のモジュールファイル名称配列（２１０１２）は、モジュールファイル名称（２１０１１）で特定されるモジュールファイルとプログラムの制御上依存関係にあるモジュールファイルのファイル名称の配列である。図２１は、先頭モジュールファイル（７０１）に付加したモジュールファイル間依存

関係情報のデータ構造を示すが、同じデータ構造のテーブルをアプリケーション実行システム（２０２）中に用意し、図１２の実行開始処理で先頭モジュールファイル（７０１）のモジュールファイル間依存関係情報（２１００）を当該テーブルにコピーするステップを加える。これにより、後述する図２２の処理でモジュールファイル間依存関係情報（２１００）が参照できるようになる。

【００６５】ダウンロード情報のデータ構造（５００）に、リーセントロードモジュールファイル名称と呼ぶ文字列型のデータ構造を加える。処理（１２００）のステップ（１２０３）と判断（１２０４）の間に、「リーセントロードモジュールファイル名称として先頭モジュールファイル名称を設定する」という処理を行うステップを挿入する。

【００６６】さらに処理（１５００）の「スタート」とステップ（１５０１）の間に、「リーセントロードモジュールファイル名称として、入力であるモジュールファイル名称を設定する」という処理を行うステップを挿入する。これにより、一番最近にメモリ上にロードされたモジュールファイルの名称がリーセントロードモジュールファイル名称として設定される。

【００６７】ダウンロードファイル決定処理（１８００）を、下記のように変更する。ステップ（１８０１）とステップ（１８０２）の間に、図２２に示すフロー（２２００）を挿入する。本フローは、ローカル変数ｎ、およびローカル配列変数Ｘを持つ。

【００６８】ステップ（２２０１）で、配列変数Ｘに、モジュールファイル間依存関係情報中の、リーセントモジュールファイル名に対応する依存関係情報エントリのモジュールファイル名称配列（２１０１２）を設定する。ステップ（２２０２）で、式「ｎ：＝１」を実行する。判定（２２０３）で、式「ｎ＜＝Ｘの長さ」を評価する。評価結果が真であるとき、ステップ２２０４に飛ぶ。評価結果が偽である場合は、ステップ１８０２に飛ぶ。判定（２２０４）で、Ｘのｎ番目のファイル名称のファイルがローカル計算機のファイルシステムのインストールディレクトリ（５０５）下にあるか判定する。その判定結果が真である場合は、ステップ（２２０６）に飛ぶ。前記判定結果が偽である場合は、ステップ（２２０５）に飛ぶ。ステップ（２２０５）で、式「ｎ：＝ｎ＋１」を実行し、ステップ（２２０３）に飛ぶ。ステップ（２２０６）で、変数ＦＩＬＥに、Ｘのｎ番目のファイル名称を設定し、ステップ（１８０８）に飛ぶ。

【００６９】上記図２２の処理により、次にダウンロードするモジュールファイルを決定する際に、最近ロードしたリーセントロードモジュールファイル名称のモジュールファイルに依存するモジュールファイルでダウンロードされていないものがあったら、それを優先してダウンロードすることになる。以上で、第１の実施形態の拡

19

張である第２の実施形態の説明を終了する。

【００７０】

【発明の効果】請求項１および３に係る発明によれば、アプリケーションを構成する複数のファイルをリモート計算機のファイルシステムからローカル計算機のファイルシステムにダウンローディングして実行するシステム、特に、入出力装置を用いたユーザとの対話的操作を多く含むアプリケーションファイルをダウンローディングして実行するシステムにおいて、ファイルダウンローディング時のユーザの待ち時間を削減してユーザの快適さを向上させることができる。特に、請求項２のように動的にダウンロードするファイルを決定するようにすれば、よりユーザの待ち時間を削減してユーザの快適さを向上させることができる。

【図面の簡単な説明】

【図１】　本発明の実施の形態で用いる計算機の構成例を示す図

【図２】　アプリケーションシステムの構成およびその動作環境を示す図

【図３】　ロードモジュールファイル管理テーブル（２０２３）のデータ構造を示す図

【図４】　ダウンロードファイル管理テーブル（２０２４）のデータ構造を示す図

【図５】　ダウンロード情報（２０２６）のデータ構造を示す図

【図６】　優先順位テーブル（２０２５）のデータ構造を示す図

【図７】　本実施形態で述べるアプリケーション実行システムが実行対象とするプログラムの構造を示す図

【図８】　モジュールファイル優先順位情報（７０８）のデータ構造を示す図

【図９】　イベントマップ（７０２）のデータ構造を示す図

【図１０】　先頭モジュールファイルのダウンロードとアプリケーション実行部の起動を説明する図

【図１１】　アプリケーション実行部によるアプリケーション

20

の実行処理のフローチャート図

【図１２】　アプリケーション実行部の実行開始処理のフローチャート図

【図１３】　優先順位テーブルの生成処理のフローチャート図

【図１４】　アプリケーション実行部のプログラム処理ブロックの実行処理のフローチャート図

【図１５】　アプリケーション実行部のモジュールファイルのロード処理のフローチャート図

【図１６】　アプリケーション実行部のアプリケーション終了処理のフローチャート図

【図１７】　ダウンロードスレッドで実行される処理のフローチャート図

【図１８】　ダウンロードファイル決定処理のフローチャート図

【図１９】　ダウンロード処理のフローチャート図

【図２０】　ＨＴＭＬのスクリプトの例を示す図

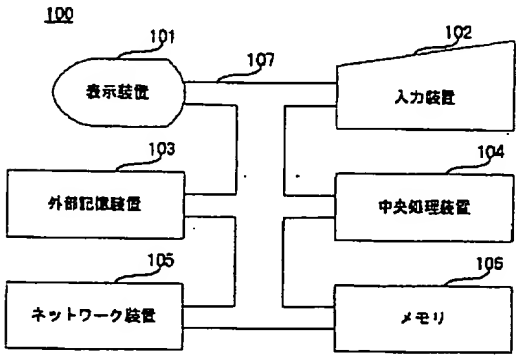
【図２１】　モジュールファイル間依存関係情報のデータ構造を示す図

【図２２】　ダウンロードファイル決定処理（１８００）への追加フローチャート図

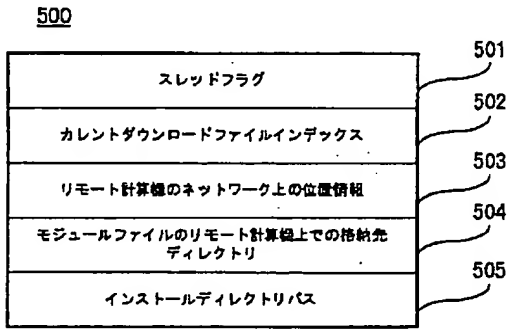
【符号の説明】

１００…計算機、１０１…表示装置、１０２…入出力装置、１０３…外部記憶装置、１０４…中央処理装置、１０５…ネットワーク装置、１０６…メモリ、１０７…バス、２００…アプリケーション実行システムの構成およびその動作環境、２０１…ローカル計算機、２０２…アプリケーション実行システム、２０３…ファイルシステム、２０４…ネットワーク、２０５…リモート計算機、２０６…サーバープログラム、２０７…ファイルシステム、２０２１…アプリケーション実行部、２０２２…ダウンロード管理部、２０２３…ロードモジュールファイル管理テーブル、２０２４…ダウンロードファイル管理テーブル、２０２６…ダウンロード情報、２０２５…優先順位テーブル。

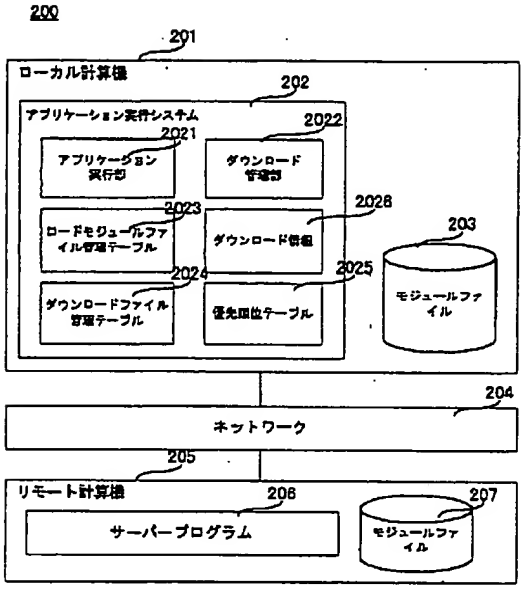
【図１】



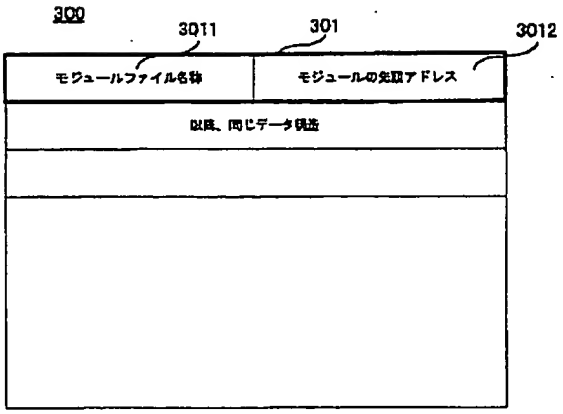
【図５】



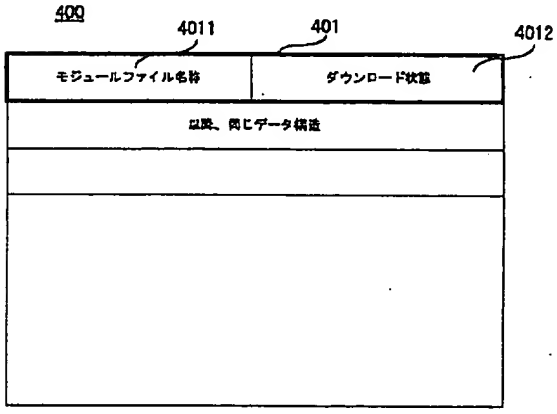
【図２】



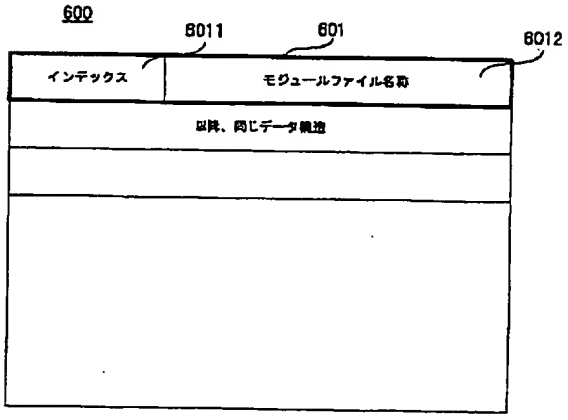
【図３】



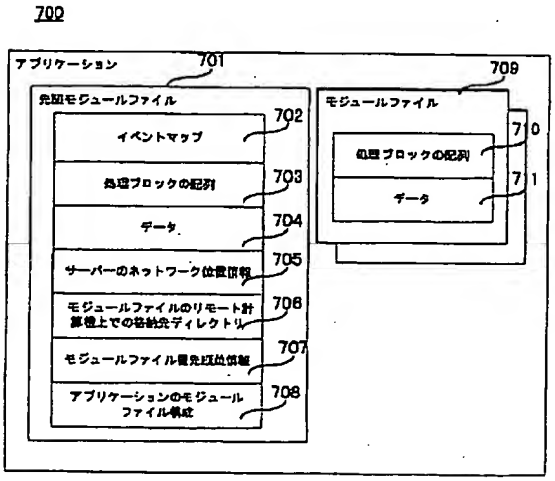
【図４】



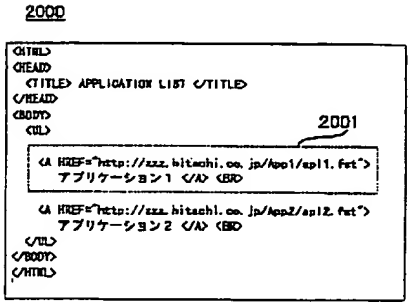
【図６】



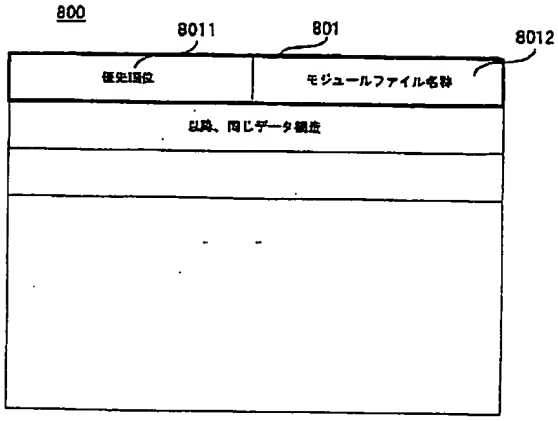
【図７】



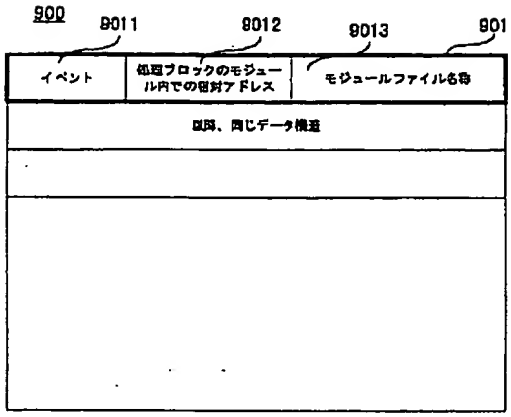
【図２０】



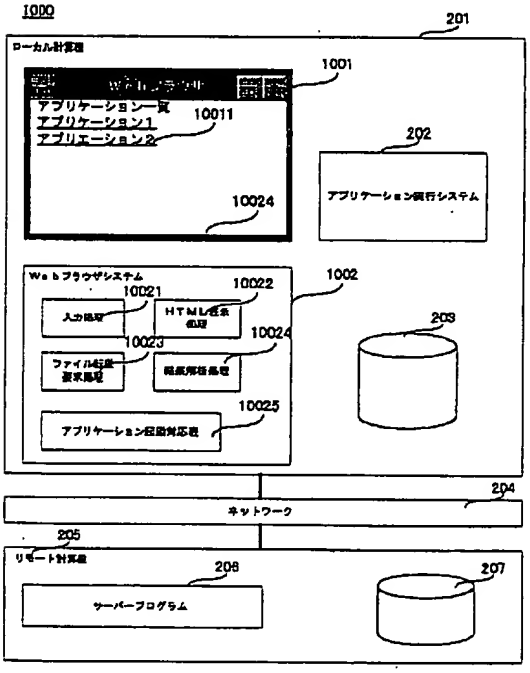
【図８】


800
8011
801
8012
優先順位
モジュールファイル名称
以降、同じデータ構造

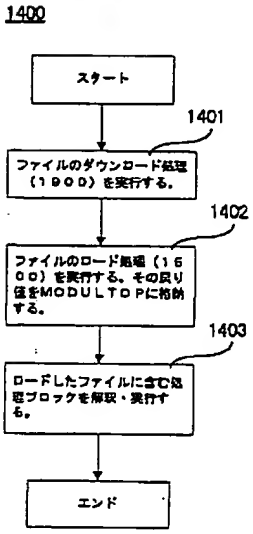
【図９】


900
9011
9012
9013
901
イベント
処理ブロックのモジュール内での相対アドレス
モジュールファイル名称
以降、同じデータ構造

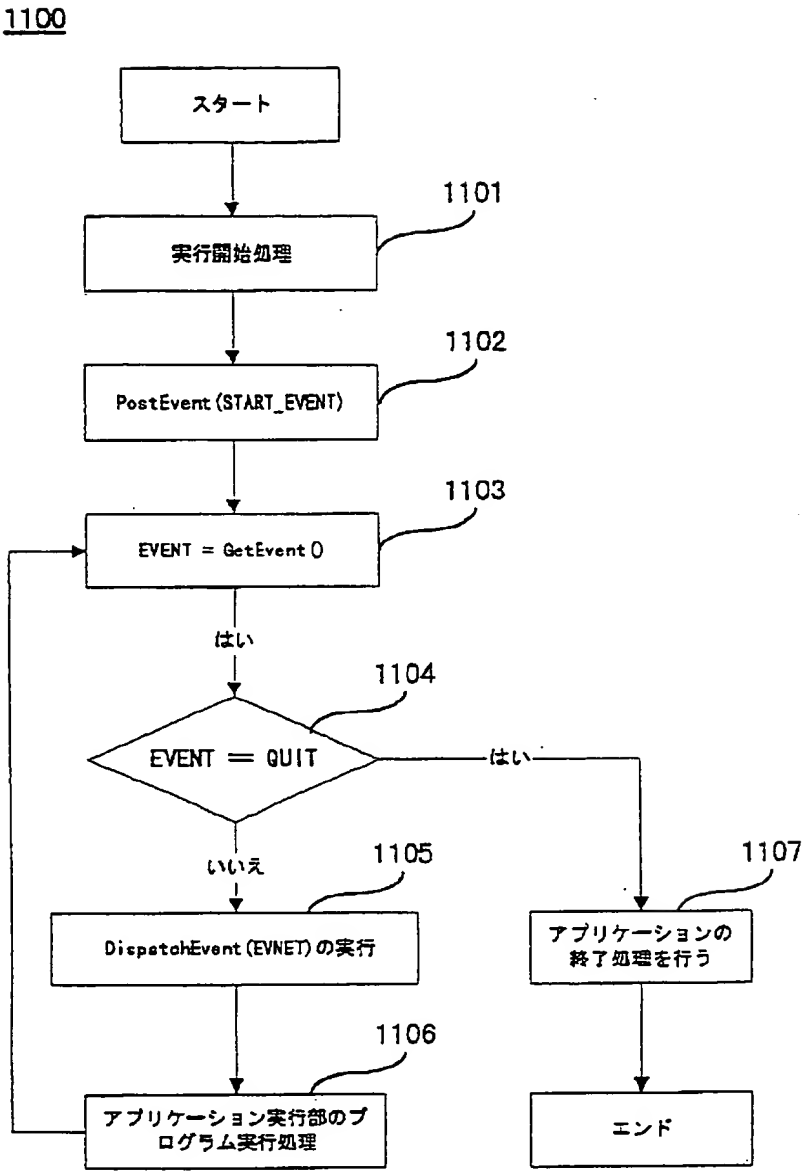
【図１０】


1000
201
ローカル計算機
1001
アプリケーション一覧
アプリケーション1
アプリケーション2
10011
10024
202
アプリケーション実行システム
Webブラウザシステム
10021
10022
1002
10023
10024
203
10025
アプリケーション起動対応表
204
ネットワーク
205
リモート計算機
208
サーバープログラム
207

【図１４】

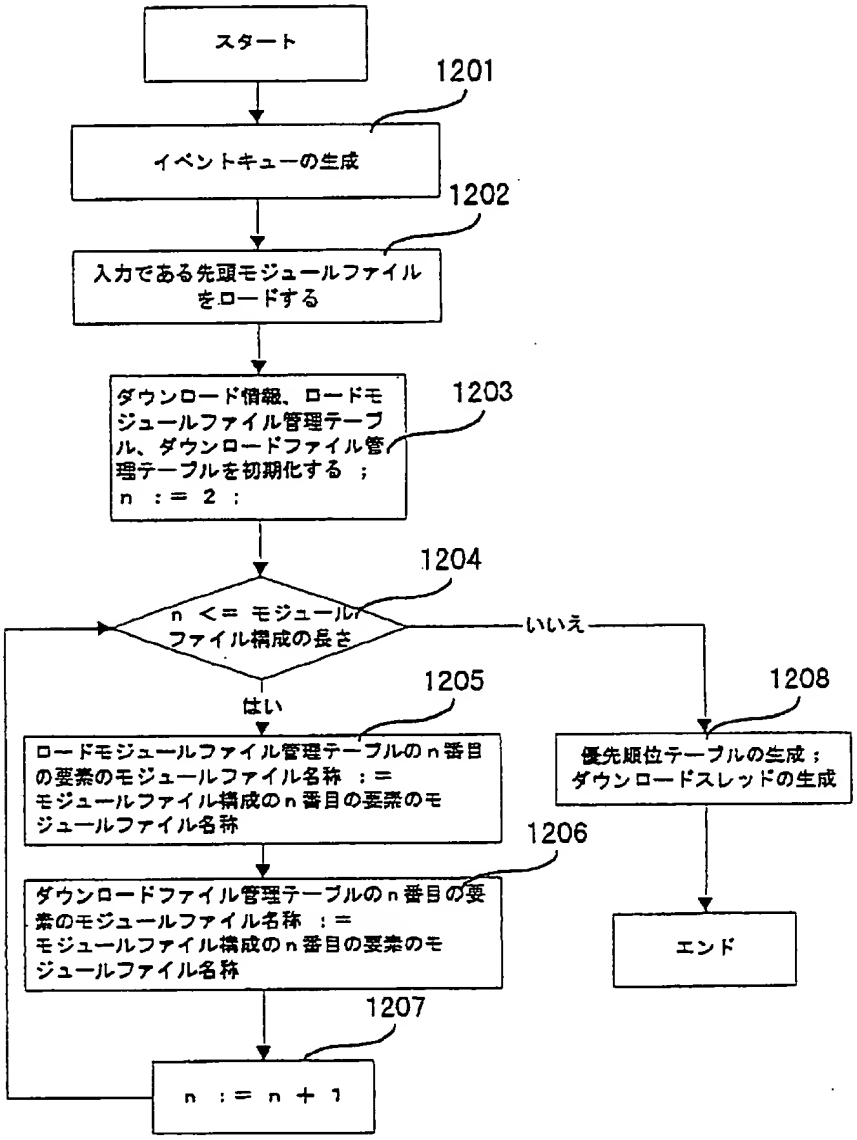

1400
スタート
1401
ファイルのダウンロード処理（１９００）を実行する。
1402
ファイルのロード処理（１６００）を実行する。その戻り値をMODULTOPに格納する。
1403
ロードしたファイルに含む処理ブロックを解取・実行する。
エンド

【図１１】

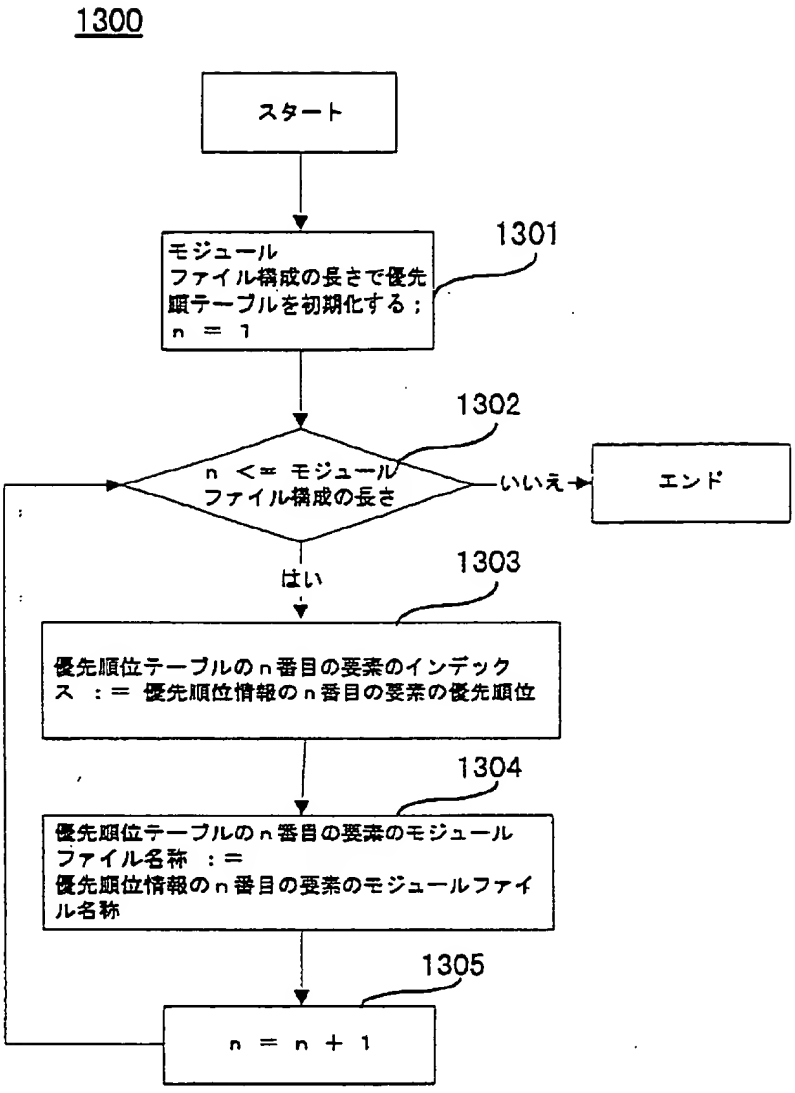

1100
スタート
1101
実行開始処理
1102
PostEvent(START_EVENT)
1103
EVENT = GetEvent()
はい
1104
EVENT == QUIT
はい
いいえ
1105
DispatchEvent(EVNET)の実行
1106
アプリケーション実行部のプログラム実行処理
1107
アプリケーションの終了処理を行う
エンド

【図１２】

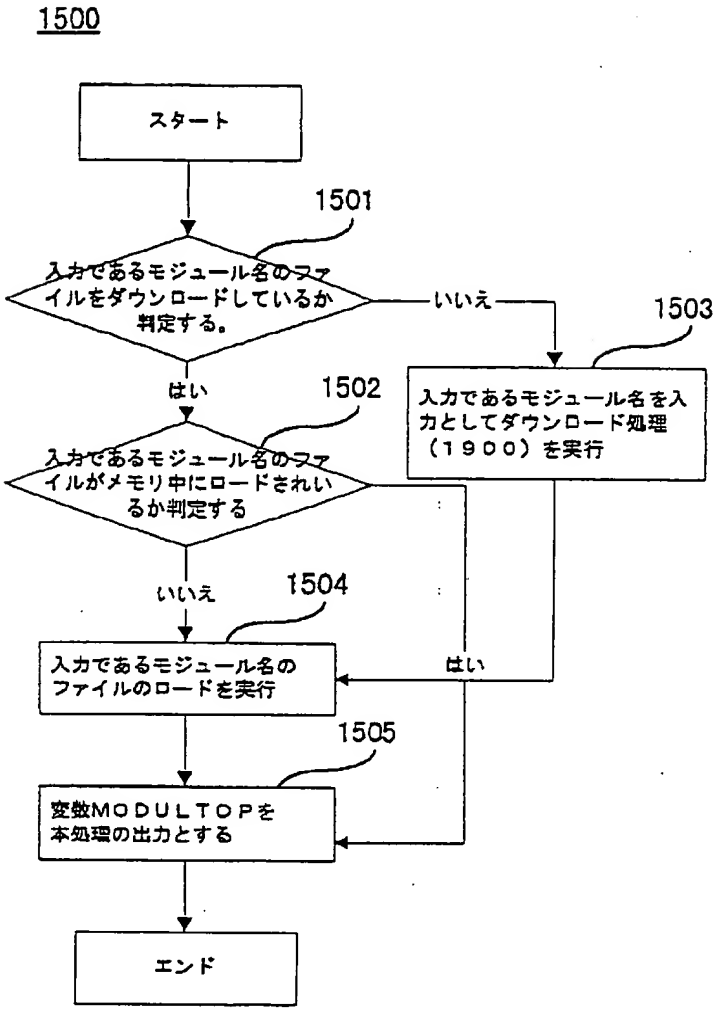
1200


スタート
1201
イベントキューの生成
1202
入力である先頭モジュールファイルをロードする
ダウンロード情報、ロードモジュールファイル管理テーブル、ダウンロードファイル管理テーブルを初期化する ；
n ：＝ 2 ；
1203
1204
n ＜＝ モジュールファイル構成の長さ
いいえ
はい
1205
ロードモジュールファイル管理テーブルのn番目の要素のモジュールファイル名称 ：＝ モジュールファイル構成のn番目の要素のモジュールファイル名称
1206
ダウンロードファイル管理テーブルのn番目の要素のモジュールファイル名称 ：＝ モジュールファイル構成のn番目の要素のモジュールファイル名称
1207
n ：＝ n ＋ 1
1208
優先順位テーブルの生成；ダウンロードスレッドの生成
エンド

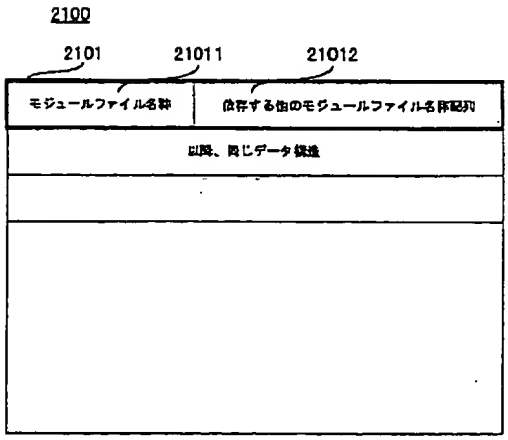
【図１３】


1300
スタート
モジュール
ファイル構成の長さで優先順テーブルを初期化する；
n ＝ 1
1301
1302
n <＝ モジュール
ファイル構成の長さ
いいえ
エンド
はい
1303
優先順位テーブルのｎ番目の要素のインデックス ：＝ 優先順位情報のｎ番目の要素の優先順位
1304
優先順位テーブルのｎ番目の要素のモジュールファイル名称 ：＝
優先順位情報のｎ番目の要素のモジュールファイル名称
1305
n ＝ n ＋ 1

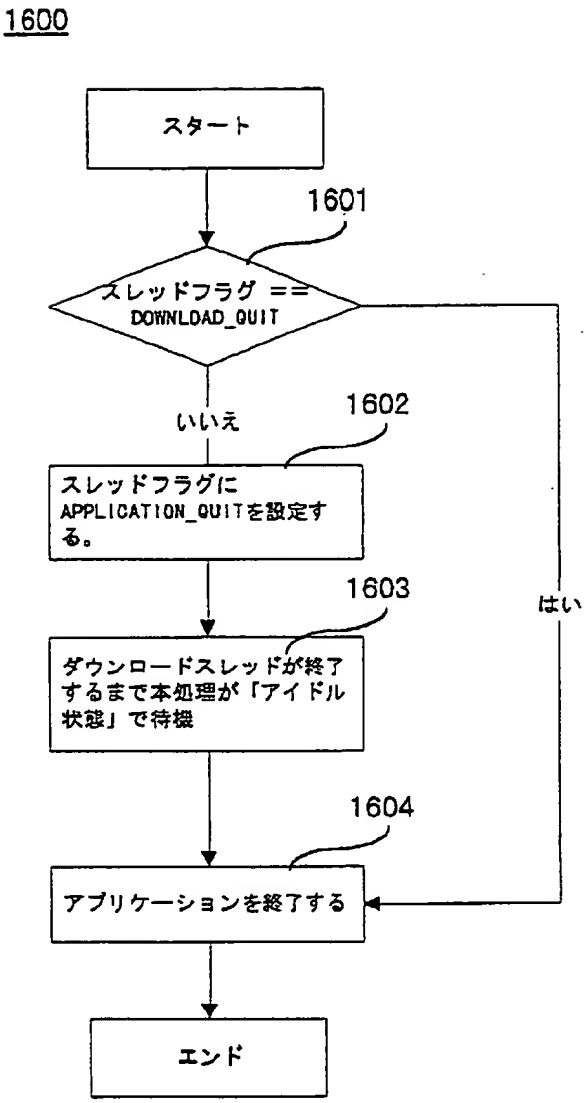
【図１５】


1500
スタート
1501
入力であるモジュール名のファイルをダウンロードしているか判定する。
いいえ
1503
入力であるモジュール名を入力としてダウンロード処理（１９００）を実行
はい
1502
入力であるモジュール名のファイルがメモリ中にロードされいるか判定する
いいえ
1504
入力であるモジュール名のファイルのロードを実行
はい
1505
変数ＭＯＤＵＬＴＯＰを本処理の出力とする
エンド

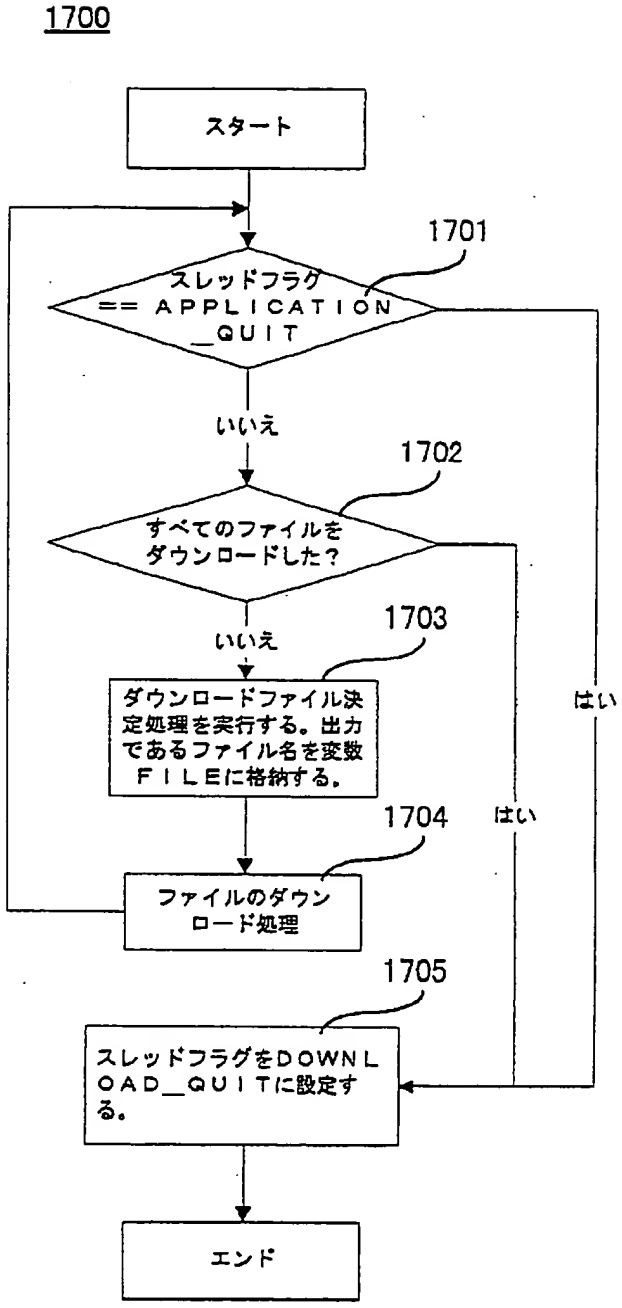
【図２１】


2100
2101
21011
21012
モジュールファイル名称
依存する他のモジュールファイル名称記列
以降、同じデータ構造

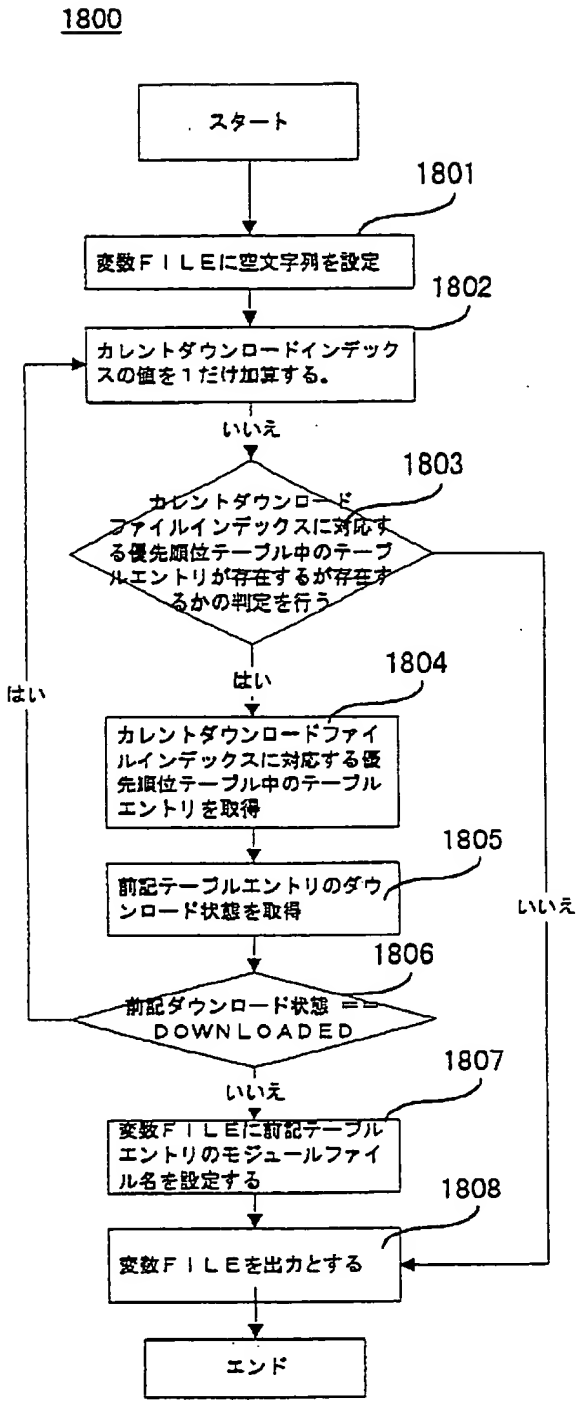
【図１６】


1600
スタート
1601
スレッドフラグ ==
DOWNLOAD_QUIT
いいえ
1602
スレッドフラグに
APPLICATION_QUITを設定す
る。
1603
ダウンロードスレッドが終了
するまで本処理が「アイドル
状態」で待機
はい
1604
アプリケーションを終了する
エンド

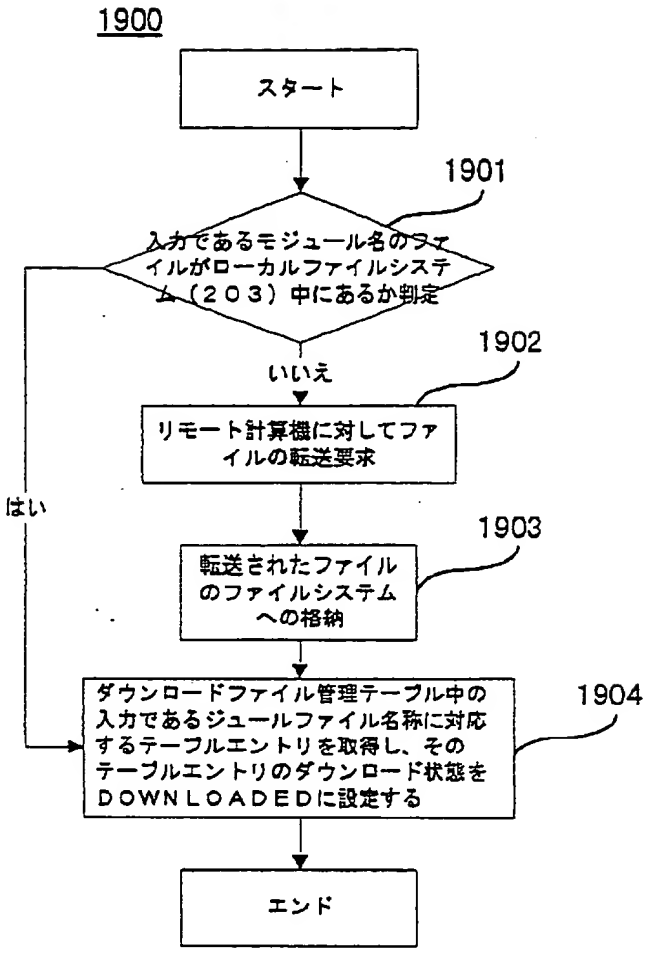
【図１７】


1700
スタート
1701
スレッドフラグ == APPLICATION_QUIT
いいえ
1702
すべてのファイルをダウンロードした？
いいえ
1703
ダウンロードファイル決定処理を実行する。出力であるファイル名を変数ＦＩＬＥに格納する。
1704
ファイルのダウンロード処理
はい
はい
1705
スレッドフラグをＤＯＷＮＬＯＡＤ＿ＱＵＩＴに設定する。
エンド

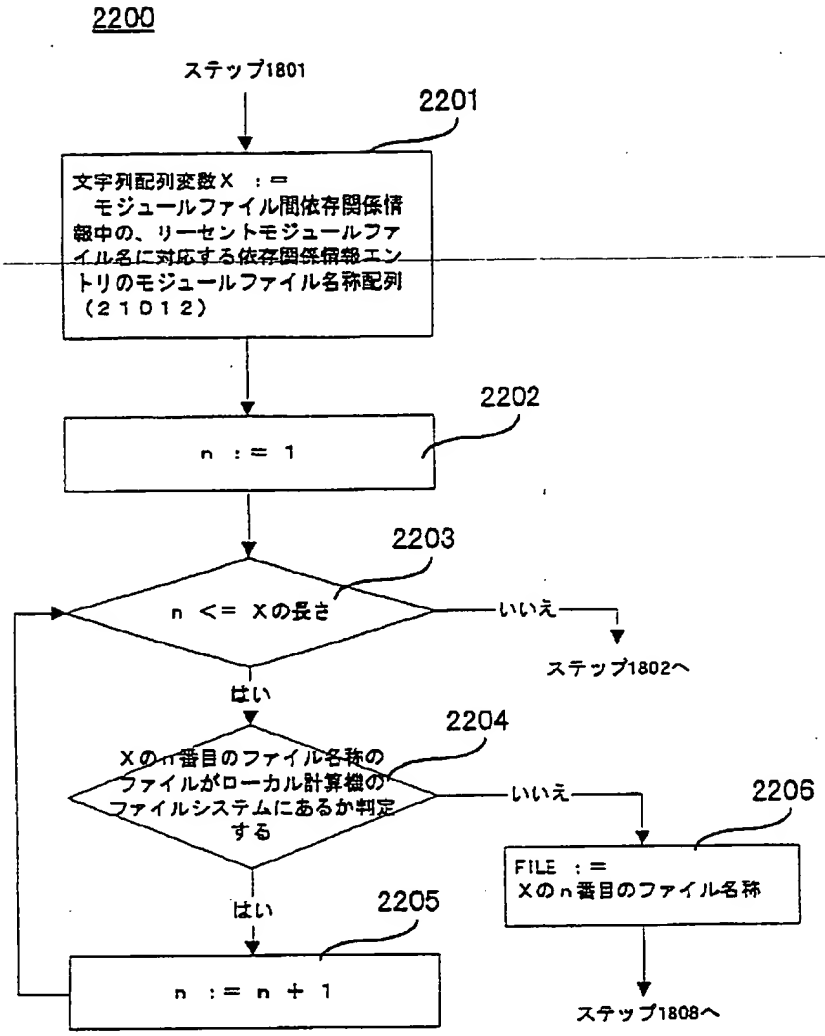
【図１８】


1800
スタート
1801
変数ＦＩＬＥに空文字列を設定
1802
カレントダウンロードインデックスの値を１だけ加算する。
いいえ
1803
カレントダウンロードファイルインデックスに対応する優先順位テーブル中のテーブルエントリが存在するが存在するかの判定を行う
はい
1804
カレントダウンロードファイルインデックスに対応する優先順位テーブル中のテーブルエントリを取得
1805
前記テーブルエントリのダウンロード状態を取得
1806
前記ダウンロード状態 == DOWNLOADED
はい
いいえ
1807
変数ＦＩＬＥに前記テーブルエントリのモジュールファイル名を設定する
1808
変数ＦＩＬＥを出力とする
エンド

【図１９】


1900
スタート
1901
入力であるモジュール名のファイルがローカルファイルシステム（２０３）中にあるか判定
いいえ
1902
リモート計算機に対してファイルの転送要求
はい
1903
転送されたファイルのファイルシステムへの格納
1904
ダウンロードファイル管理テーブル中の入力であるジュールファイル名称に対応するテーブルエントリを取得し、そのテーブルエントリのダウンロード状態をＤＯＷＮＬＯＡＤＥＤに設定する
エンド

【図２２】



フロントページの続き

(72)発明者　本田　由里
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(72)発明者　嶋崎　康一
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内